BEAM: Integrated Technology for Autonomous Self-Analysis

Ryan Mackey

Amir Fijany Mark James

Han Park Michail Zak

Ultracomputing Technologies Group

Section 367

BEAM:

Beacon-based
Exception
Analysis for
Multimissions

An integrated, on-board or off-board method of data analysis for fault detection, anomaly detection, and prognostics

Combines physics-based models, state models, statistical models, and sensor data

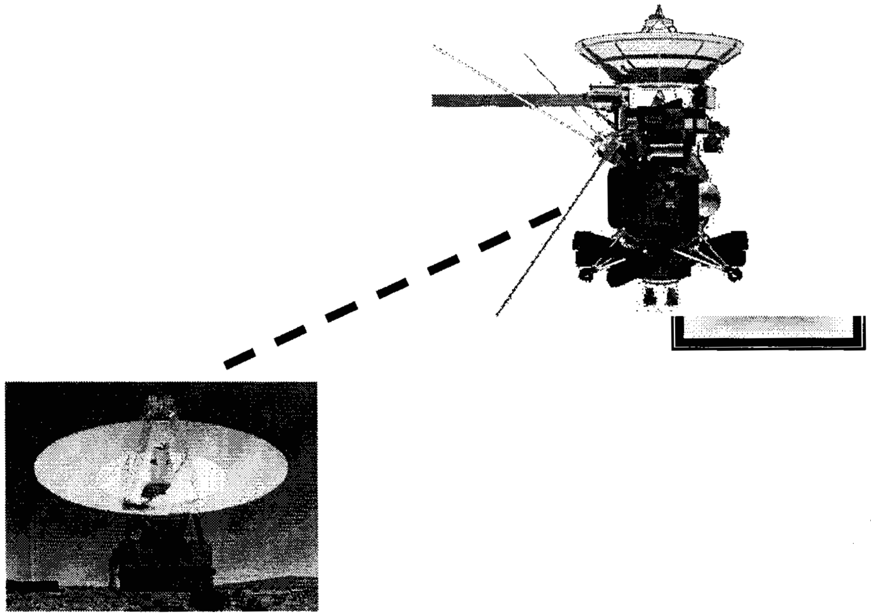
Technology developed at JPL under IPN-ISD

Ancient History: Automatic vs. Autonomic

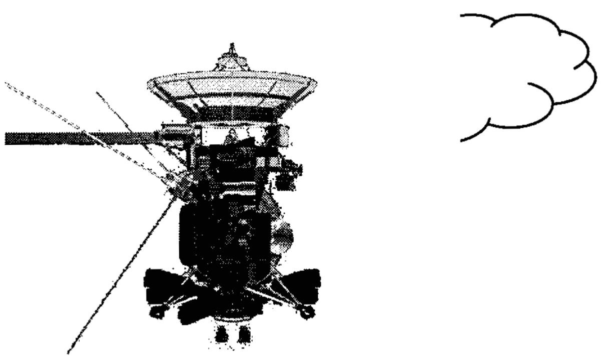
- ***Automatic Systems*: Acting without conscious volition or control**
- ***Autonomic Systems*: Behavior in a manner indistinguishable from conscious control**

- **Key distinction is "conscious" vs. "unconscious" control**
- **What do we mean by conscious control?**
- **How do human operators do their jobs?**
- **When is conscious control necessary?**
- **How difficult is it to apply?**

How to apply judgement?



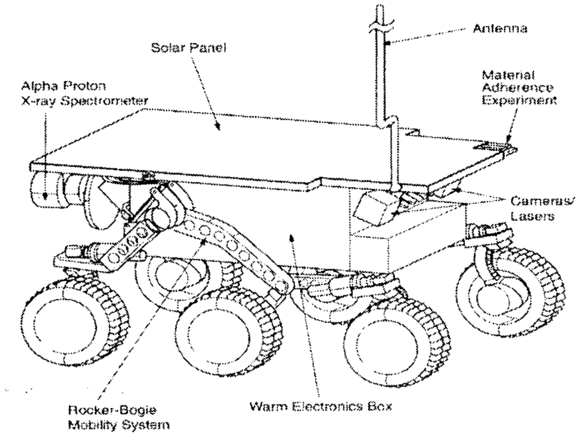
- ***Ground operators:***
  - ◆ Time of response important
    - Critical periods of operation
    - Rapid science phenomena
  - ◆ Visibility of data
    - Downlink and data bus constraints
    - Problems with sensors



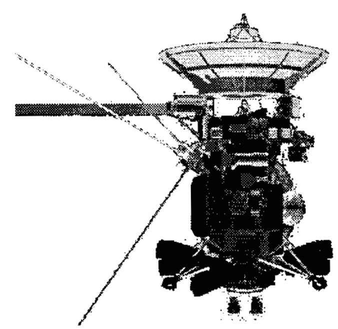
- ***Existing on-board software:***
  - ◆ Complexity of failures
  - ◆ Sensor coverage
  - ◆ Computing resources
  - ◆ Confidence in autonomy

On-board Software Resources

- Simple spacecraft: probably none
  - Even Sojourner had capability for simple sensing and fault detection



- Complex spacecraft:
  - Receives commands
  - Executes on a clock
  - Takes and stores measurements
  - Monitors for known faults
  - Sends measurements to ground



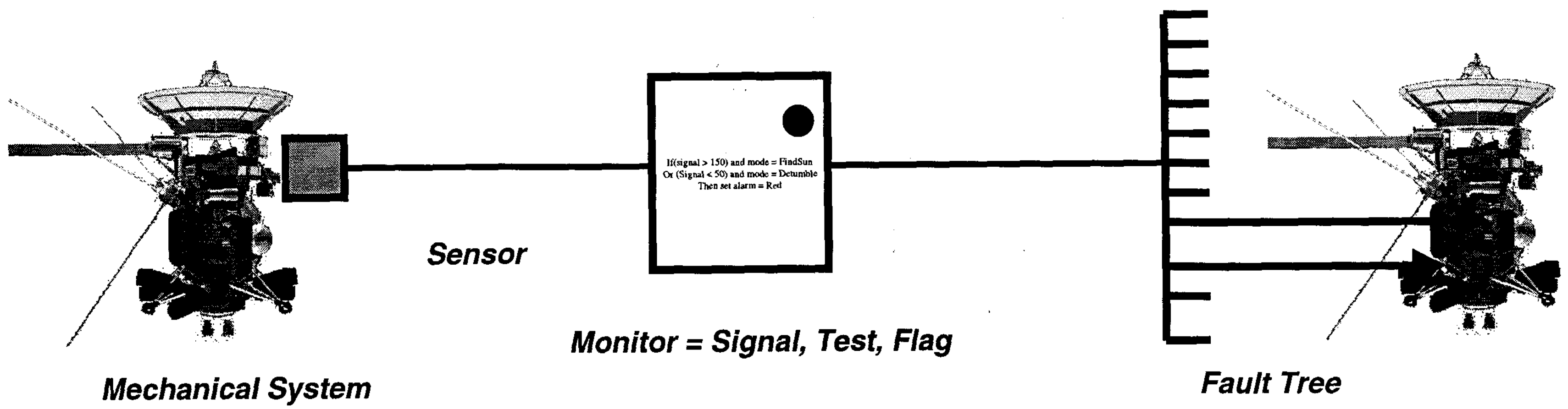
Definitions

- *Failure:* Physical damage to the system causing degradation or inoperability of system functions
- *Fault:* A measurable (not necessarily measured) misbehavior in in the system (i.e. a symptom)
- *Discrepancy:* A measured difference between expected and actual system behavior

---

- Not all failures cause faults
  - Systems that are not in use are usually impossible to sense
- Not all faults are due to failures
  - Interactions between components
  - Accidental command problems
  - Environmental change beyond assumptions
- Not all faults cause discrepancies
  - Faults may counteract
  - Smaller disturbances than expected
  - Mitigating adaptations (self-adapting control)
- Not all discrepancies indicate faults
  - Sensor failures
  - Data collisions
  - Noise misinterpretation

Traditional Fault Detection Methods



- **Typical fault detection:**
  - **Design-time identification of specific symptoms**
  - **Sensor selection to detect those symptoms**
  - **Monitors with specific rules to analyze sensor data**
  - **Listing of possible faults for each monitor indication**
- **Difficulties:**
  - **Expensive!!**
  - **Hard to reconfigure**
  - **Low resiliency to changes**

- **How to take advantage of autonomy?**

Software Wish List

- ***Self-Monitoring:***
  - Intelligence to separate source faults and secondary effects
  - Method of detecting problems that are not caused by failures
  - Detection of things that "look funny"
  - Reduce and classify data sent to the ground
  - Provide a means to react to things that "look funny"
  - Consideration of interactions between subsystems
  - Faster, better cheaper!

- **Goals:**
  - Simplify process of data analysis
  - Enhance spacecraft safety and availability
  - Improve spacecraft flexibility
  - Reduce fault protection design effort
  - Enable riskier missions

Scientific Approach

- **What is the best way to apply judgement to data analysis?**
  - Spacecraft under control can be treated as a complex laboratory experiment
    - Scientist observes response to environment
    - Excites system through commands and state transitions
    - Spacecraft sensors produce measurements and indications about the system

- **Experimental technique:**
  1. Understand what the system is asked to perform
  2. Determine qualitative results and observations
  3. Retrieve quantitative measurements about the system
  4. Examine quantitative data for interesting features
  5. Test for known phenomena
  6. Compare data to physical understanding
  7. Focus on exceptional behavior as determined by past experience
  8. Predict future behavior of the system

Fault Detection Parallel

- Software Architecture:
  - Understanding system commands and status variables
    - State model of system
    - Interprets commands and predicts internal state of spacecraft
  - Testing for known faults
    - Status reports can be checked against state model
    - Discrepancies in status and quantitative data checked by expert system
  - Adding physics knowledge of the system
    - Physics models of subsystems compared to data (theoretical knowledge)
    - Statistical models compared to residual data (experience)

The Anomaly Hypothesis

- *Anomaly:* An unexpected event in the system, either captured by sensors, status information, or indirect observation

  - Anomalies are a superset of faults
  - Anomalies do not always imply that a fault has occurred
  - Some failures trigger anomalies but not faults
    - Degradation or nonlinearities
    - Incomplete understanding of system (false anomaly)
    - Environmental interaction
    - Trending to failure (prognostics)

  - Anomalies *do* imply that improvement is needed to fault detection
  - Software maintenance case
    - Anomalies can be ranked to reduce engineering data
  - Autonomy case
    - Broad-class anomaly detectors can be partially reasoned upon
  - Performance metrics: False-alarm rates and missed detections
    - Difficulties: Subjectivity of false-alarm rates
    - Other alternatives: Total availability, total number of safings, etc.

A Specific Detection Method

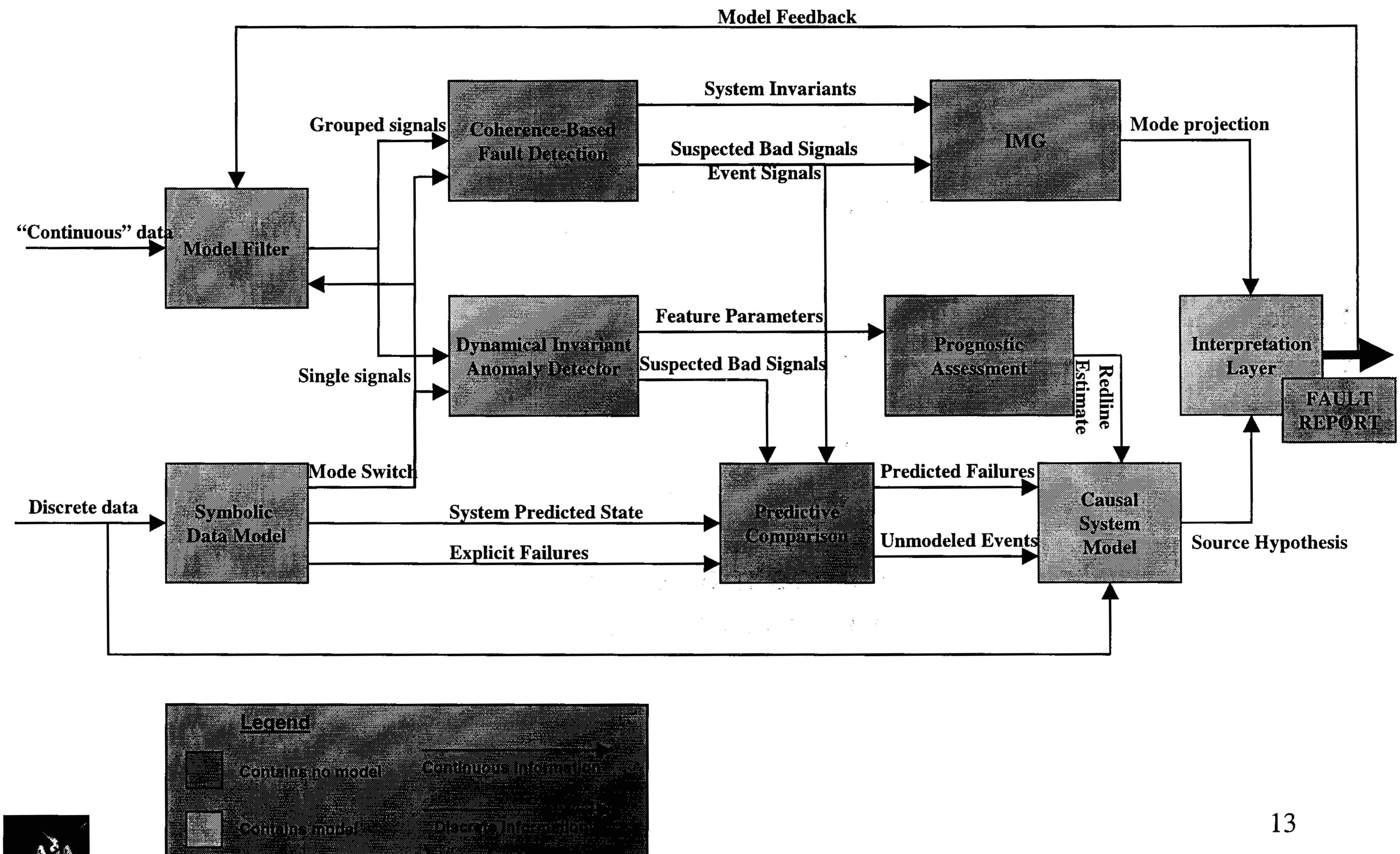
Experimental Technique:

1. Understand what the system is asked to perform
2. Determine qualitative results and observations
3. Retrieve quantitative measurements about the system
4. Examine quantitative data for interesting features
5. Test for known phenomena
6. Compare data to physical understanding
7. Focus on exceptional behavior as determined by past experience
8. Predict future behavior of the system

BEAM Components:

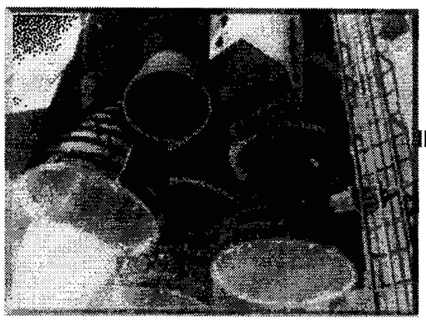
1. Symbolic Data Model (SHINE)
2. Sensor Data (synchronized) (Model Filter, SDM)
3. Sensor Data (conditioned) (Model Filter)
4. Signal Processing (Coherence Detector, Feature Extraction)
5. Predictive Comparison (SHINE)
6. Combine with physics model (Gray Box)
7. Statistical modeling (Coherence Detector, Feature Extraction)
8. Prognostics (Predictive Assessment)

BEAM Architecture
Top Level Block Diagram

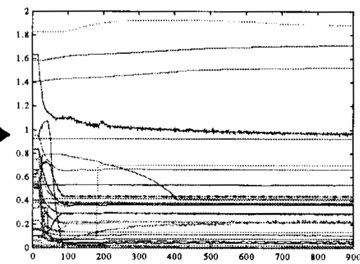


Application: Tools for Space Shuttle Main Engine (SSME)

Space Shuttle Main Engine



Raw Sensor Data



BEAM Compact Fusion Operation for Diagnosis and Prognosis

Task Objective:

- Develop diagnostic and prognostic tools
- Detect subtle anomalies and degradation in SSME ground tests
- Develop real-time capability
- Prove BEAM for on-board implementation

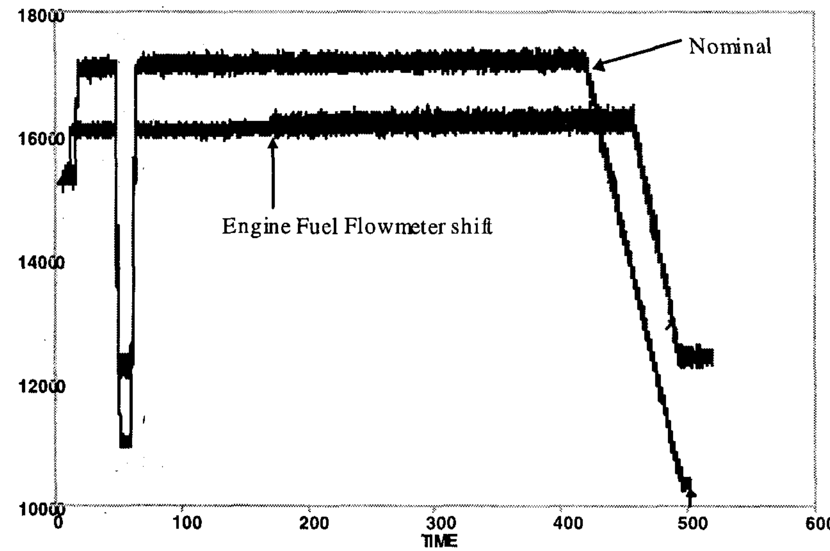
Relevance:

- Rapid, automatic analysis of large data sets
- Robust fault detection and isolation
- Drastically reduce cost of SSME operations and maintenance

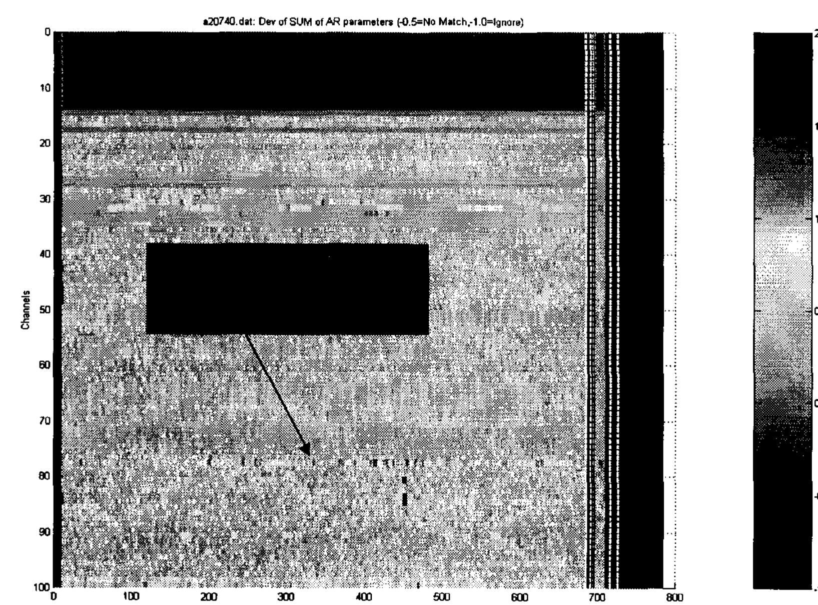
BEAM

SSME Anomaly Detection

- **Example: Fuel Flowmeter Shift**
  - **Unmodelable phenomenon**
  - **Degraded engine performance**
  - **Mechanical damage**
  - **Not detectable with current diagnosis tools**



- **Method:**
  - **Train detector on nominal data**
  - **Apply detector during run or as part of post-analysis**
  - **Console tool operated by MSFC staff**

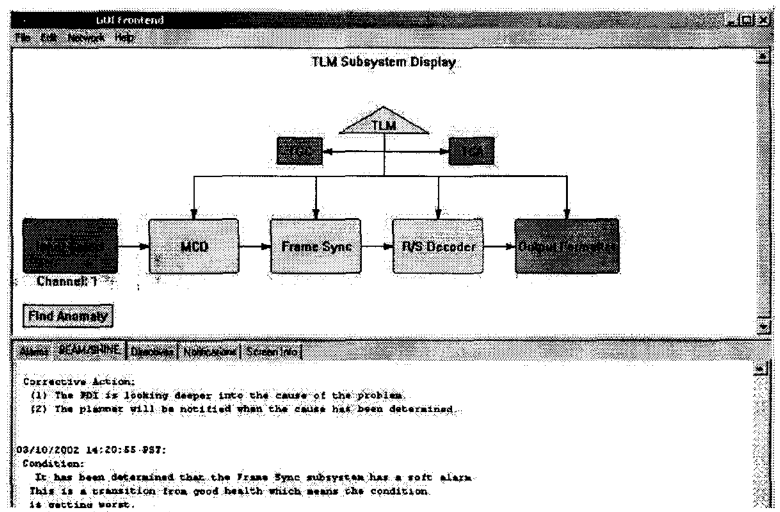


- **Reference:** ***Analysis of Space Shuttle Main Engine Data Using Beacon-Based Exception Analysis for Multimissions*, (submitted) IEEE Aerospace Conference 2002**

BEAM

Application: DSN Common Automation Engine

- **Current DSN Operations:**
  - Operators must watch several screens during antenna track
  - CAE intelligently summarizes to a single screen
- **CAE Fault Detection and Identification GUI Features:**
  - System state summary
  - Hierarchical representation
  - Equipment status
  - "Find Anomaly" button
  - Reports
  - Fault detection logging
  - Charts
  - Event timelines



BEAM Technology Conclusion

- Autonomy provides a different framework for spacecraft design
- Comparison to scientific method gives us a useful perspective

- Provide complete diagnostic and prognostic assessment
  - Comparable performance to human operators or pilots
  - Robust response to “novel” conditions
  - On-board or off-board implementation
  - Alert ground systems to anomalies prior to landing or encounter
- Use *all* sources of system information
  - Maximize information from existing sensors
  - Include all state information, commands, state models
  - Include all available physical models
  - Quantify information redundancy
- Permits partial or full autonomy for complex aerospace systems
  - Improve reliability or eliminate need for scheduled maintenance
  - Reduce operating or logistics footprint

Backup Slides

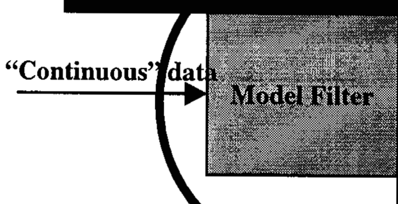
BEAM Architecture Details

Future Research

Model Filter: "Gray Box" Component

- **Separates time-correlated sensor data**
  - Known [redacted]
  - Stationa[redacted]
  - Linear [redacted]
  - Non-line[redacted]

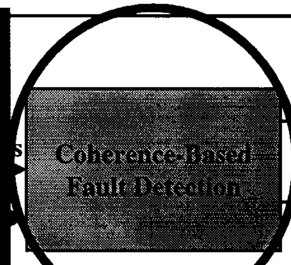
[redacted]



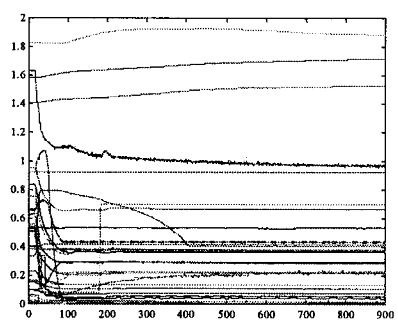
[redacted]

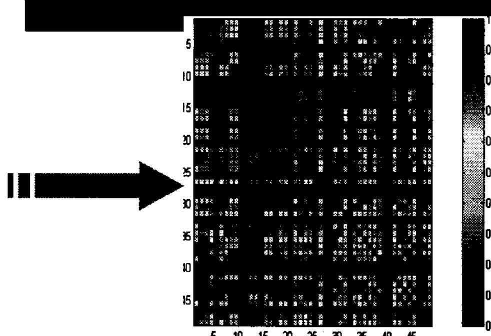
BEAM

[redacted]



[redacted]

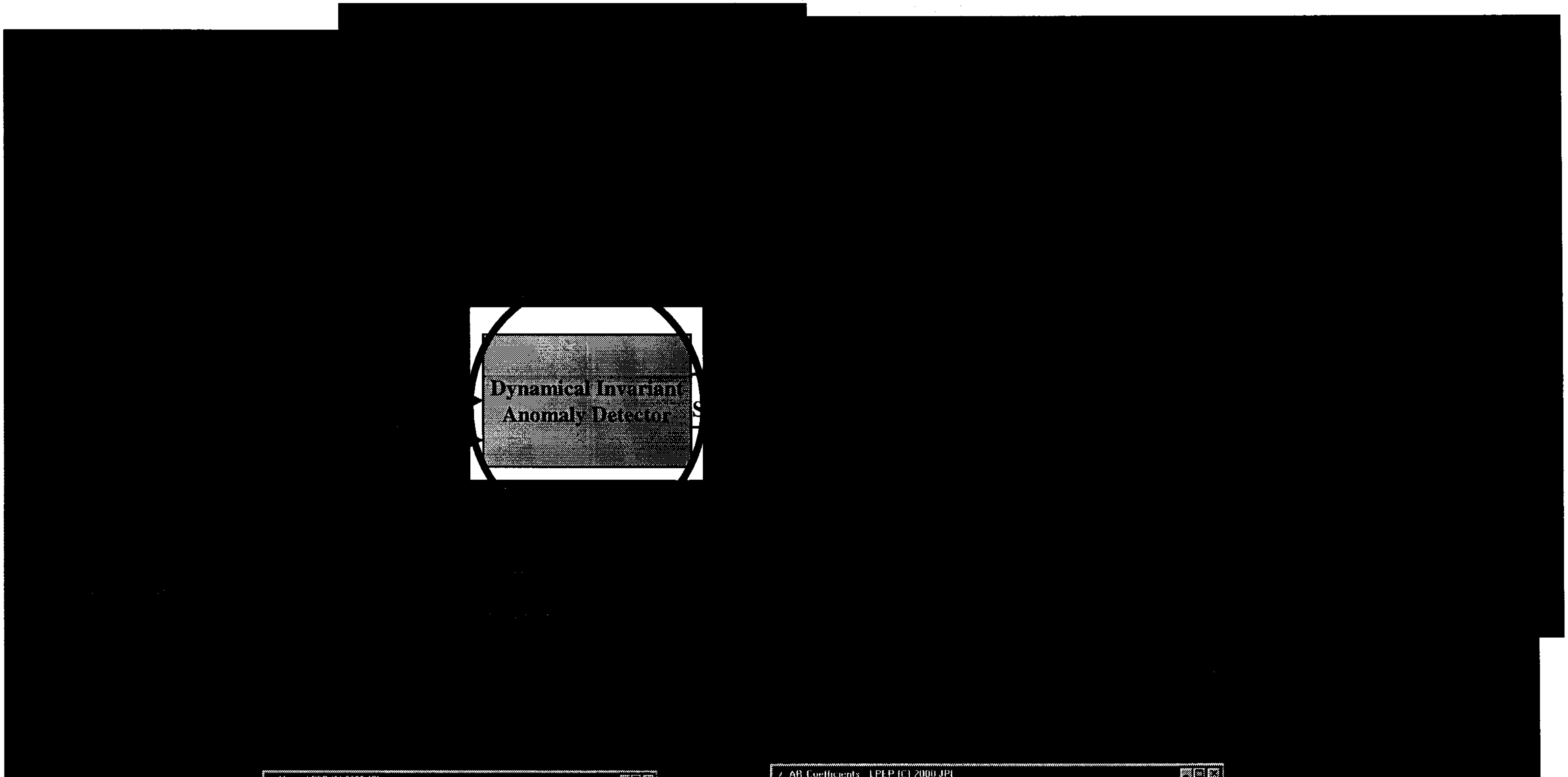




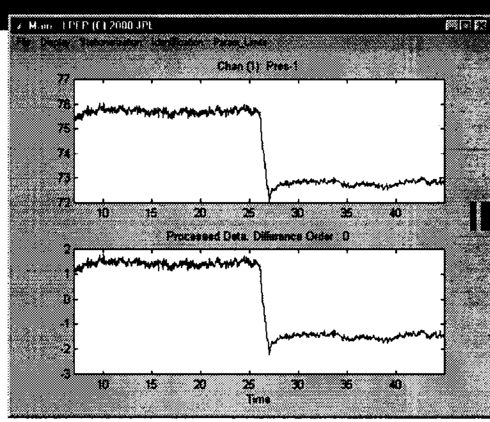
Raw Sensor Data

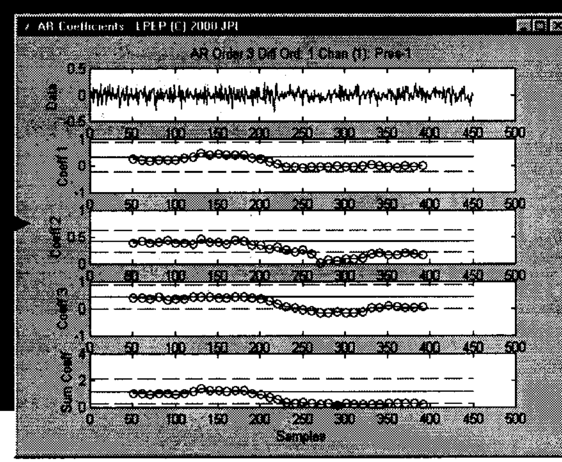
Subsystem C [redacted]
(Evolving in Time)

BEAM



Raw Sensor Data

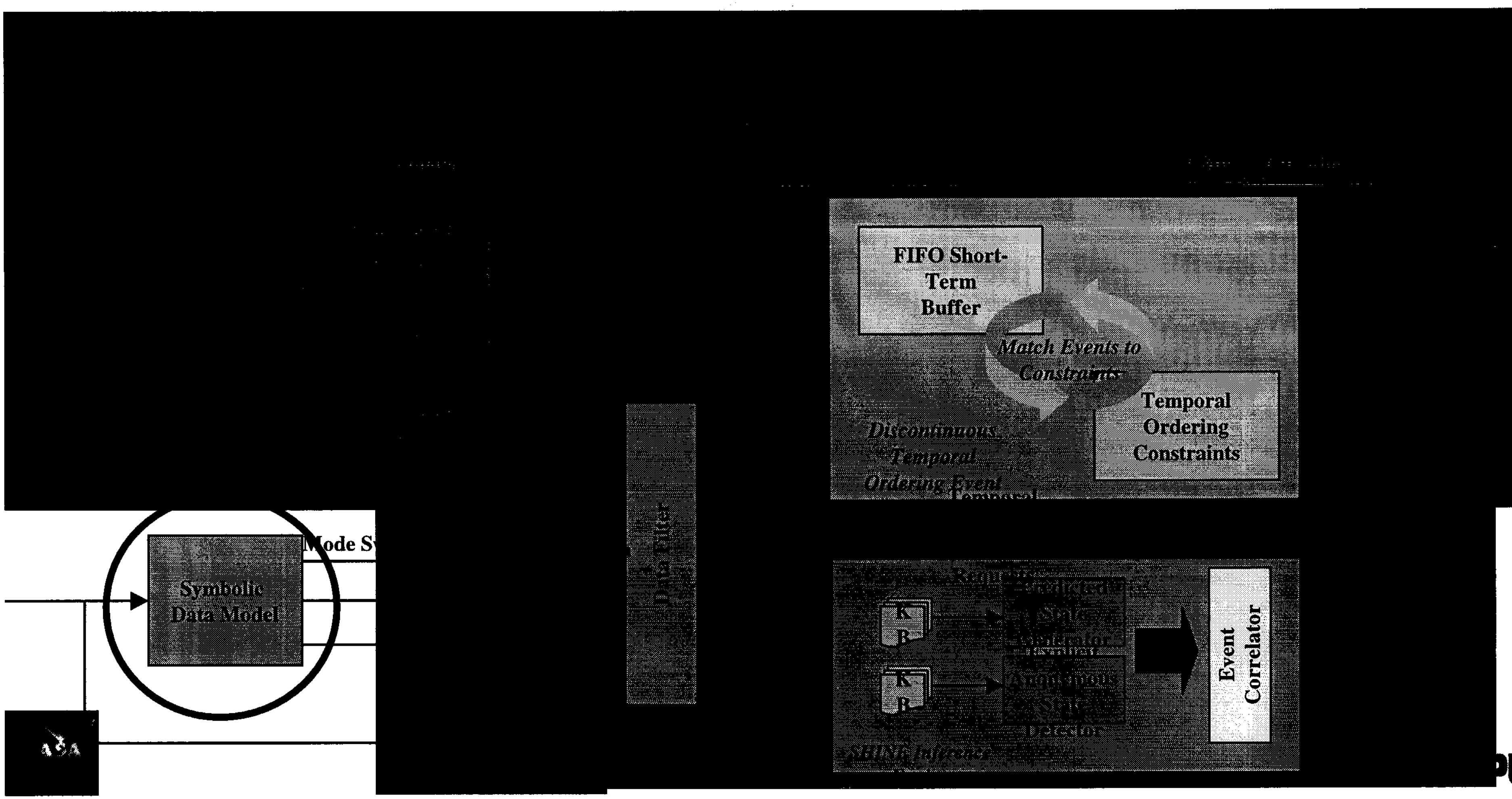




JPL

Symbolic Data Model

- Considers all discrete signals from the system
- Detects and enumerates state mismatches and explicit failures
- Identifies operating mode of the system
- Predicts state of system components



Future Technology: Model Reconstruction

- **Improve accuracy of gray-box technique**
  - **Enhance deterministic component of Gray-box when:**
    - **Physical models do not exist**
      - *Too complex for direct modeling*
    - **Real-time performance is required**
      - *More efficient computations*

- **Construct dynamical models from sensor observations**
  - **Proper Orthogonal Decomposition (POD) modeling**
    - **Create low-order dynamical models using:**
      - *POD mode extraction*
      - *Galerkin projection*
  - **Dynamical networks with topological self-organization**
    - **Network with well-organized tensor structure**
      - *Attractors and basins can be easily incorporated and controlled*
      - *Structure is similar to many physical systems*